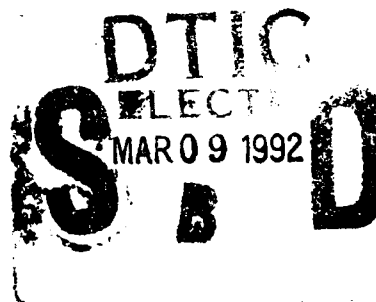AD-A248 652

THE PARA RADIATION EFFECTS CURRENT SIMULATOR

Bharat Bhuva
Sherra Kerns

Mission Research Corporation
1720 Randolph Rd SE
Albuquerque, NM 87106

January 1992

DTIC
ELECT
MAR 0 9 1992
S B D

Final Report

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

92-06015

PHILLIPS LABORATORY
Directorate of Space and Missiles Technology
AIR FORCE SYSTEMS COMMAND
KIRTLAND AIR FORCE BASE, NM 87117-6008

92 3 ... ...

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely Government-related procurement, the United States Government incurs no responsibility or any obligation whatsoever. The fact that the Government may have formulated or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication, or otherwise in any manner construed, as licensing the holder, or any other person or corporation; or as conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

This report has been authored by a contractor of the United States Government. Accordingly, the United States Government retains a nonexclusive royalty-free license to publish or reproduce the material contained herein, or allow others to do so, for the United States Government purposes.

This report has been reviewed by the Public Affairs Office and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nationals.

If your address has changed, if you wish to be removed from our mailing list, or if your organization no longer employs the addressee, please notify PL/STE, Kirtland AFB, NM 87117-6008, to help us maintain a current mailing list.

This report has been reviewed and is approved for publication.
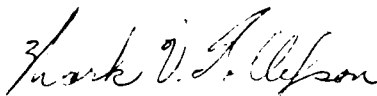

R. VIRGIL OTERO
Project Officer

RUSSELL R. HERNDON, GM-14
Chief, Electronic Systems Br

FOR THE COMMANDER

MARK V. TOLLEFSON, Lt Col, USAF
Chief, Space Electronics Div

DO NOT RETURN COPIES OF THIS REPORT UNLESS CONTRACTUAL OBLIGATIONS OR NOTICE ON A SPECIFIC DOCUMENT REQUIRES THAT IT BE RETURNED.

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE January 1992	3. REPORT TYPE AND DATES COVERED Final, 8 Mar 90 - 8 Oct 91

4. TITLE AND SUBTITLE THE PARA RADIATION EFFECTS CURRENT SIMULATOR	5. FUNDING NUMBERS C: F29601-89-C-0014 PE: 612120 PR: 1430 TA: 01 WU: 04
6. AUTHOR(S) Bharat Bhuva Sherra Kerns	

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Mission Research Corporation 1720 Randolph Rd SE Albuquerque, NM 87106	8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Phillips Laboratory Kirtland AFB, NM 87117-6008	10. SPONSORING/MONITORING AGENCY REPORT NUMBER PL-TR--91-1071

11. SUPPLEMENTARY NOTES

Subcontractor for this report was Vanderbilt University, Nashville, TN 37235

12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution unlimited.	12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

The Parametric Analysis of Radiation effects software development is given the general title PARA. This report describes the development of algorithms and source codes for the simulation of radiation effects on CMOS ICs. The project concentrated on the simulation of total dose effects and the ways to establish/predict the operational lifetime and radiation tolerance of ICs. The switch level simulator incorporates the capability to assign bias dependent post-irradiation drive parameters to transistors within a microcircuit and to calculate propagation delays based on those parameters. This permits test vectors to be assigned based on worst case post-irradiation propagation delays.

14. SUBJECT TERMS Radiation Effects Software, IC Radiation Operating Performance Prediction, IC Failure Mode Identification, Switch-level Simulation Algorithms			15. NUMBER OF PAGES 42
			16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT SAR

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. Z39-18 298-102

CONTENTS

CONTENTS (CONCLUDED)

iv

1.0 INTRODUCTION

The Parametric Analysis of Radiation effects software development efforts at Vanderbilt University are given the general title PARA. The PARA efforts were started at North Carolina State University by Dr. Sherra Kerns. The initial version of PARA was developed to help design engineers, test engineers, and other interested engineers determine the vulnerability of a circuit to total-dose radiation. At first, the software was developed in a supporting role to verify the experimental results. However, with the ever increasing cost of radiation experiments, the roles of the software and experiments were reversed. The PARA software is being used to predict the operating performance of ICs in radiation environments and the simulation results are verified through experimentation.

However, as with all other software packages, the simulation results from PARA must be thoroughly verified. Also, to make a software useful to the engineers, the developers must present the software through a user-friendly interface. If the software is wonderful but very difficult to master, very few engineers will use it.

Below is a brief description of PARA along with the shortcomings of the previous versions and the proposed improvements. This is followed by the accomplishments of the past year and future possibilities. In the discussions that follow, no distinction is made between the numerous earlier versions of PARA. The earlier codes consisted of research versions and were never released. The general description of earlier PARA shortcomings summarizes the totality of earlier research results.

1.1 PARA

PARA is a switch-level tool capable of identifying the principle failure mode caused by the design methodology for a CMOS digital IC. The field-oxide-leakage failure is not considered a part of this failure mechanism as it is controlled by the fabrication process parameters and not the circuit design. The principle failure mechanisms for the PARA simulation were mainly divided into three categories: static failure, dynamic failure, and power-related failures.

Static failure is considered a stuck-at type of failure which occurs only at a node in the circuit. This type of failure causes the node voltage to be out of the digital logic range for a 1 or a 0 (i.e., the voltage for the node is neither HIGH nor LOW but somewhere in between). The subsequent logic circuits do not recognize this voltage and the overall circuit operation fails. This type of failure is caused by the increased leakage current of n-channel devices combined with decreased current driving capabilities of p-channel devices. This type of performance degradation is highly bias dependent.

Dynamic failure occurs over a path consisting of a sequence of gates rather than a single node. Due to the decreased p-channel pull up current and increased n-channel leakage, less current is available for a node to charge the capacitance. This results in a longer delay for the signal to rise to its final value. The cumulative effect of individual increase in propagation delays at nodes can amount to a significant increase in delay over a path causing a timing failure. This is especially true in case of synchronous circuits where timing is a critical issue. Such failures are highly dependent on the bias conditions during and after irradiations.

Power-related failure refers to the case when the leakage through the circuit nodes causes the standby current to increase beyond the system specifications, giving rise to a failure. This type of failure, thus, involves all gates on an IC. The large number of such gates on a VLSI circuit indicates that the individual dependence on bias conditions for each gate has no overall effect on the final degradation.

PARA uses switch-level simulation algorithms along with RC-delay estimation techniques to speed up the simulation process with acceptable accuracy. PARA has been tested on many digital logic circuits consisting of simple logic gates. However, digital circuits where logic functions are implemented through the use of pass gates, mainly used for generating XOR or XNOR functions, cannot be modeled by the switch level algorithms in PARA due to its analog nature (accurate delays through such a gate can be obtained only through analog circuit simulators). Switch-level simulation techniques do not accurately estimate the delays through such a gate, and hence PARA, in its current form, cannot simulate such a circuit.

The simulation results from PARA have been verified through exhaustive simulation through accurate circuit simulators, such as SPICE. The final results were within 10 percent of each other. Circuit simulation times were much smaller for PARA than for SPICE, with the actual improvement factor dependent on the size of the circuit. For small circuits containing less than 10 transistors the improvement is small. For large circuits having several hundred transistors improvements of 3 orders of magnitude have been observed. However, the simulation speed and accuracy were still not optimal.

For a detailed descriptions of the algorithms used in PARA and other details, refer to the published literature.

1.2 PROPOSED TASKS FOR 1990-1991

The main object for the PARA project for 1990–1991 was to improve the software in whatever manner possible and prepare it for a public release. The main tasks for the year 1990–1991 were decided as follows:

1.2.1 Decrease Overhead Processing

To make PARA useful, many small things had to be incorporated in the PARA software. For example, PARA accepted node names instead of numbers. This increases overhead as all the node names must be compared to find a match. If node numbers were accepted, only two numbers need to be compared. Comparing two strings of characters (a node name) takes much more computer time than comparing two integers. Similar trivial things made PARA really slow for very large circuits with a large number of nodes. Processing overheads had to be reduced.

1.2.2 Add Resistive Elements

PARA, in its original form, did not accept any interconnect resistances. As PARA translates all the transistor network into an RC-network, the interconnect resistances were assumed to be much smaller than the device resistances and were ignored. However, it was suggested that addition of interconnect resistive elements will improve the accuracy. The interconnect resistance was assumed to be the resistance between logic gates and not within the logic gates.

1.2.3 Accept Hierarchy of Cells

Previous versions of PARA did not accept hierarchical description of circuits to facilitate creation of a data base. However, generating a flattened list of devices for large files from software requires large chunks of memory and that was deemed inconvenient. It was decided to develop a preprocessor to accept hierarchical description of circuits. As PARA operates on individual devices, this hierarchical description has to be converted eventually into a flattened listing. The proposed preprocessor was expected to accomplish this.

1.2.4 Improve Standby Current Estimation Algorithm

Originally PARA accepted only one value of leakage currents for n-channel devices to perform the power-related failure analysis. However, with this approach, some accuracy was lost due to the different leakage mechanisms involved. For example, there are three different leakage mechanisms involved for radiation environments: channel leakage, edge leakage, and back-gate leakage for Silicon on Sapphire (SOS) and Silicon on Insulator (SOI) technologies. Modifications proposed for PARA were to accept all these leakage mechanisms and give an accurate estimation of standby power requirements.

1.2.5 Design of Test ICs

For verification purposes, a proposal was made to design a medium-scale integration (MSI) level IC and have it fabricated at one of the commercial houses. This finished product was to be used for testing and verification purposes.

1.2.6 Enhanced User Interface

PARA was developed without any user-friendly interface. It was decided that an improved user interface would encourage engineers to adapt it into the design and test cycle faster. An X-Window based, mouse-driven interface for PARA and all other related software was proposed.

1.2.7 Path Properties

It was proposed that path properties for all the worst-case paths be reported to the user. The path properties are the statistics of the path under consideration, such as number of NAND, NOR and inverter gates, a path with maximum number of NAND gates, etc. Such data could be used to identify many things.

1.2.8 Data Base Accessibility

PARA was expected to be incorporated into the existing design structure. To accomplish that, software engineers required a window into the PARA data base to acquire whatever information is needed. It was proposed that a window into the data base similar to the one developed by Cadence Design Systems Inc. software be developed.

These tasks were generated with consultation from U.S. Army Harry Diamond Laboratories (HDL) and Mission Research Corporation (MRC) personnel. These tasks were carried out at Vanderbilt University. The software was developed on UNIX based systems. The programming language used was C. The software was made platform independent to avoid any problems later on.

1.3 SUMMARY OF RESULTS

All of the above mentioned tasks were completed except the Enhanced User Interface. In addition to these tasks, other tasks were also carried out to make PARA useful to the radiation effects community. These tasks were development of a BiCMOS circuit simulator, a statistical circuit simulator, a test vector generator for total dose testing, and a circuit synthesizer.

4

With all these improvements, PARA is ready for distribution. The aim for the next year, following the present effort, is to develop a user interface combining all of the software to generate a generalized radiation effects simulator. The simulation algorithms for all these simulators should be able to handle large circuits very easily. However, no circuit simulator can be considered complete in this age of rapid advancements. PARA can be improved in many ways. However, the first priority should be to develop a user interface to attract users to PARA. Many good software programs have failed due to an inadequate user interface. PARA will face the same fate unless it is made very user friendly. Another improvement should be incorporation of acceptance of analog-type circuits. There are many ways PARA can be improved for the next generation of ICs. However, it is complete in its current form.

2.0 ALGORITHMS

PARA converts a given circuit into a resistive and capacitive network. The MOS devices are converted into resistances and parasitic capacitances are added later on to model the delays and power requirements. Brief descriptions of the modeling of MOS devices as resistances and the actual simulation algorithms follow. The next section describes the shortcomings of the last version of PARA and the improvements made to overcome those shortcomings.

2.1 ALGORITHMS

2.1.1 Resistance Estimation

PARA estimates the ON postirradiation resistances of devices for all cases (ON during irradiation, OFF during irradiation) by running SPICE simulations on a chain of four inverters. The time elapsed between the input of an inverter rising/falling to 2.5 V and the output of the inverter falling/rising to 2.5 V is found to be the most suitable time constant for resistance calculation for circuits with VDD − VSS = 5 V. Circuits using other supplies can be analyzed with minor modifications. In order to suppress the effect of internal node and gate capacitances, large capacitances (10 pF) are added at each node. These capacitances determine the time constants, rather than the internal device capacitances. The time constants when divided by these capacitances result in the average resistance of the transistor (p or n, as the case may be).

2.1.2 Static Failure

The static failure can be simulated by estimating the minimum p-channel current sourcing capabilities and the corresponding maximum n-channel current sinking capabilities at the node. The ratio of p-channel drive divided by the n-channel drive provides an estimate of the vulnerability of the node for such a degradation under extreme conditions. The smaller the ratio, the more vulnerable the node is. Any other leakage paths, such as through pass transistors need to be added to the denominator for calculating the ratio.

By sorting the nodes in ascending order of ratios, the nodes with decreasing vulnerability can be obtained. The minimum pull up current can also be translated to maximum pull up resistance for switch level simulations. Minimum pull down resistance can also be used instead of the maximum pull down current. Maximum pull up resistance is usually offered by the longest path of series p-channel transistors from the node to Vdd. The bias conditions that provide the worst value of the ratio are established so as to provide minimum pull up and maximum pull down currents, i.e. p-channel devices to be turned

6

OFF and n-channel devices to be turned ON. This requires that inputs to the p-path (and n-path) devices be at HIGH logic level.

2.1.3 Dynamic Failure

This involves the evaluation of maximum timing delays through all the circuit paths. Delays at individual nodes in the paths are calculated by taking maximum RC delays. The maximum resistance can be obtained by extracting the maximum pull up resistance path as in the previous case for the signals rising from LOW to HIGH. For the falling signals, the maximum resistance pull down path through n-channel transistors is used. This normally is the maximum number of series n-channel transistors from the node to ground for properly designed circuits. For calculating the maximum delays, the delays need to be calculated for all possible input bias conditions during irradiation (while the parameters are shifting) and after irradiation (after the parameters have shifted). The four cases that form an exhaustive set of bias conditions, for a particular path, are the following.

- Input HIGH during irradiation, going from LOW to HIGH after irradiation

- Input HIGH during irradiation, going from HIGH to LOW after irradiation

- Input LOW during irradiation, going from HIGH to LOW after irradiation

- Input LOW during irradiation, going from LOW to HIGH after irradiation

The paths that show maximum delays form the critical paths and are most vulnerable to dynamic failure. The values of path degradation are, in general, a function of the rate of irradiation.

PARA first of all identifies all the input nodes. The circuit is traversed for any one of the input nodes until an output node is reached. This is a depth-first approach where all fanouts are neglected and only one signal path is considered. After reaching the output node, PARA calculates the delays associated with this path by starting at the output gate and moving backward. This process is repeated until all the paths associated with this input node are taken care of. At this point, only the worst-case paths for this input node are stored in memory. The same process is repeated for all the input nodes. As the number of nodes and signal paths in fast RAM is small, the revised PARA is much faster than the previous versions.

While the above discussions are in terms of positive logic, PARA can also be applied to negative logic. The only difference between positive and negative logic is the notation used for HIGH and LOW levels. PARA operates with the two voltage levels (i.e., 5 V

and 0 V) and does not care what the designer calls HIGH and LOW. Currently PARA assumes positive logic while generating out in terms of HIGH and LOW states. If negative logic is used, the designer can interchange the HIGH and LOW state labels in the output results.

2.1.4 Chip-Level Failure

The chip-level failure calculation involves the determination of maximum leakage at nodes with logic level HIGH. The leakage at nodes with logic LOW do not have significant leakage. In addition, all the nodes are never HIGH at the same time. Therefore randomly selected nodes are needed for average standby current calculation. The leakage can be estimated by extracting the maximum leakage path through the n-channel transistors connected between the node and ground. Monte Carlo strategy is used to estimate the average standby current for the whole chip. The algorithm takes the following form. Fifty percent of the nodes with logic level HIGH are selected randomly and the standby current is estimated. This process is repeated a number of times to arrive at an objective average estimate.

The final average value is obtained from all average leakage values obtained through the above repetitions. The individual leakage currents of a device are categorized into edge leakage and channel leakage. Edge leakage depends inversely on the length of the device while channel leakage depends on the width of the device. Provisions for a third type of leakage, back-gate leakage, for SOS/SOI devices is also provided. If SOS/SOI technology is not used, this leakage parameter is set to zero.

2.2 THE NEED FOR IMPROVEMENT

The need for improvements in PARA version 2.0 was obvious in many areas. These have been described briefly in the first section. Here a detailed description for each of the problems along with the solutions used is presented.

2.2.1 Decrease Overhead Processing

The need for improvement in this area of PARA version 1.0 was obvious from the start. The old algorithms for PARA were the ones developed at North Carolina State University. These were mainly developed to demonstrate the feasibility of such a simulator. The complexity and unstructured nature of the source code made this first version difficult to distribute to various vendors and manufacturers. A second version, PARA 2.0, was proposed with a machine independent software package for distribution. In addition, for PARA 2.0, a data base was designed to suit the algorithms in terms of accessibility, efficiency, modularity, and speed. This ensured the ease of any modifications for the

inclusion of technologies other than CMOS, e.g. Bipolar or nMOS. However, the memory requirements of PARA were ignored in the hope that bigger computers would have enough memory to handle large circuits. This assumption was proven false as the circuit size increased exponentially while the computer RAM memories did not. The insufficient RAM availability in computers caused PARA to use virtual memory (virtual memory means PARA ran out of fast RAM and had to write data onto the disk). Whenever PARA needed data from virtual memory, it had to wait for the computer to access the disk and transfer data to the RAM. This slowed PARA down for extremely large circuits. The data base requirements had to be changed to stop PARA from using up all of the fast RAM.

The second main problem with PARA was that it allowed names (instead of numbers) for a given node. This increased the flexibility of PARA from a designer's point of view but increased the overhead processing by orders of magnitude. As PARA uses R-C networking during simulation, it looks for the nodes which are connected by comparing all node names and selecting the ones with identical names. In a circuit with 5000 node names (a very small circuit by today's standard) and 100 signal nodes, PARA would have to compare 500,000 names. As name comparison takes longer than number comparison (number comparison checks to see if two numbers are equal by checking their binary equivalent: for comparing names, the binary equivalent of each character in the name must be checked), PARA started to become slower for large circuits with thousands of node names.

For the latest version of PARA, it became evident that to avoid high memory requirements, the data base would have to be changed. The most memory was taken up by the variables storing all the characteristics of paths and gates. This variable was changed and the most memory consuming data were removed. The data which were deleted stored the characteristics of each and every path in the circuit (such as the nodes, gates, delays, etc). The new version of PARA calculates all these variables dynamically (meaning they are obtained whenever needed). This increases the processing overhead but reduces the memory requirements by orders of magnitude enabling PARA to remain in fast RAM.

The second improvement was made in the way PARA stores the data base. Originally, all the nodes were stored as simple variables. To find a node with a given name, PARA compared all the node names to identify the gate to which the node belonged. This was a very time consuming operation. For the latest version of PARA, hash table techniques were used. For this technique, all the node names are passed through a hash function which converts the name of the node into a number. This number is used as an address to store the data. Whenever a node is required, its name is passed through the hash function to obtain the address at which the data are stored. This is a faster operation than the string compare method used in the earlier versions.

PARA now has the following variables.

- The Device Array

  All the MOS transistor declarations in the input file are stored in this array. Each of the components of the array represents one transistor and all its attributes. The attributes constitute identifiers for the four transistor terminals, transistor dimensions and SPICE model parameters to be used while computing device characteristics.

  Current versions of PARA take the four terminals as the drain, gate, source and substrate. PARA does not differentiate between SOS, SOI or Bulk technologies. The specific technology is implicit in the model parameters supplied by the designer.

  The model parameters are simply those required by the designers preferred version of SPICE. PARA itself does not have a transistor model and only uses the resistance values from a SPICE analysis of a delay chain.

- The Resistor-Capacitor Arrays

  PARA stores all the resistors and capacitors in individual arrays. An individual element of these arrays would be a capacitor or a resistor with its attributes. The attributes here comprise identifiers for the two terminals and the component value.

- The Model Parameters

  The model parameters provided by the user are placed in a separate array. Each of the transistors has a pointer to one of the elements of the model array. Each model parameter assumes a default value unless it receives a definite value from the circuit description file or ".res" file.

- Signal Node List

  The list of signal nodes forms a key data structure on which PARA operates frequently. A signal node essentially is an output node of a gate. For digital circuits, this node is the only node of interest in a gate, except for the input nodes which are assumed to form the output nodes of other gates. The signal nodes are extracted from the circuit description and saved in an array. The array element, therefore, is a single signal node with all the associated information. The kind of information, besides the name, appended to each node constitutes the following.

  - The path of n-channel transistors

    Every CMOS gate has a series of n-channel transistors forming paths from signal node to ground. PARA stores all identifiers for those n-channel transistors which form a path between the signal node and ground in an exact topology

10

as they appear in the circuit. Dynamic allocation of memory is used to accommodate as many transistors as appear in a path, the limit determined only by the memory available.

   − The path of p-channel transistors

For every n-channel transistor in a gate, a p-channel transistor exists. Therefore, PARA stores paths of p-channel transistors exactly in the same fashion as n-channel paths.

   − Capacitance

The total capacitance at a node is the sum of all the gate capacitances, all the node (junction) capacitances, and all the externally connected capacitances attached to that node. These capacitances are not to be confused with the 10 pF capacitances used for estimating the device resistances. These capacitances, used for calculating delays at individual circuit nodes, are the actual capacitance values obtained from the circuit description.

• The Hash Table

The hash table is a block in memory specially reserved for storing data by PARA. The size of this block can be changed on the fly as required. A hash function is also generated which converts a given string of characters into a number which falls within the size of the memory block. The hash function can be any function. A simple hash function just adds the ASCII values of all the characters in the given string to produce a number. This number is converted to a modulo n number where n is the size of the memory block. A more complicated hash function was used in this simuation.

The hash functions may have collision problems, meaning, two character strings may have the same hash address. In this case, the conflict must be resolved and the data stored to facilitate fast retrieval. One way to achieve this is to use pointers and cellars. A cellar is a block of memory locations reserved for such collisions and this block resides within the hash memory block. If two names happen to have the same address, one of them is stored in the address determined by the hash function and a pointer is attached to that memory location indicating that another name with the same address is stored at the location.

For the PARA hash tables, the drain, source, and gate nodes of each transistor were stored. At each address, three memory locations are reserved, one for the gate, one for the drain/source, and one for the pointer. Only one location is used for drain/source as they are treated as equivalent. The memory location actually stores the number of the transistor to which the nodes belong. For example, if the gate node of transistor # 14 has hash address of 127 and the drain node of transistor

11

34 has the same address, then the first eight bits at memory location 127 will contain the number 14, the second eight bits will contain 34, and the remaining 16 bits will contain either the address of the conflicting data (if a collision occurs) or nothing (if there are no conflicts).

These hash tables are used during the simulation process to identify the signal flow for a given input. The identification of the next stage is done dynamically. For the given input node, all transistors with the gate connected to each are identified through the use of gate-hash table. Next, a signal node list is used to identify the output of each logic gate. This output node now becomes the input node for the next stage and the same process is repeated until a circuit output node is reached. This process of dynamic identification of the next logic gate is very fast and the storing of all the signal paths is not necessary easing the memory requirements and keeping PARA within RAM.

2.2.2   Add Resistive Element

In today's ICs with a million transistors, the problem of interconnect resistances is becoming increasingly evident. This is mainly caused by the lengths of the interconnects. In PARA 2.0, the interconnect resistances were assumed to be much smaller than the device resistances, and therefore were ignored. However, for future technologies, and for some of the present day designs, the interconnect resistances may not be negligible. For these reasons, it was decided that interconnect resistances should be one of the elements. The interconnect resistance will be used for failure analysis.

PARA 3.0 allows interconnect resistances in between two logic gates. However, interconnect resistances inside a logic gate are not allowed. The resistances outside are merged with the resistance values of the devices inside the logic gate and a final resistance value generated to obtain a delay.

As these interconnect resistances are on the order of hundreds of ohms and the transistor resistances modelled in the thousands of ohms, the simulation results will not be influenced very strongly by interconnect resistances. However, for cases where it may contribute significantly to signal delay, the new models should prove adequate enough for simulation.

2.2.3   Accept Hierarchy of Cells

For Very Large Scale Integration (VLSI) and beyond technologies, the requirement that a circuit be described in a flattened format is too restrictive. The generation of the circuit description will take a long time. However, as PARA operates on individual devices, a flattened description is needed at some point in the simulation process. A separate routine

12

acting as preprocessor for circuit descriptions has been developed. This preprocessor flattens the design and prepares a flattened file for PARA. As many software packages require this flattened format, a separate routine seemed to be the best approach. This routine has been developed using general guidelines for cell hierarchy so as to facilitate its usage by all of the software without running into any technology-related problems.

The preprocessor first takes all the sub-circuits in a given design and flattens them by adding a character at the end of all the node names. For example, if a node in a sub-circuit is named out1, it will be changed to out1a when flattened for one level of hierarchy. If the same cell is encountered second time, the node name will be changed to out1b. The whole alphabet is used repetitively, i.e., after encountering 26th instance of a cell, the next one will contain out1aa. This type of flattening is very fast and does not require a lot of computing time. The final flattened file is passed on to whichever software is being used or stored in a file for future use.

2.2.4   Improve Standby Current Estimation Algorithms

The previous versions of PARA estimated the standby current through the use of normalized leakage currents for each device. This leakage current was not further divided into different categories. However, for many technologies, such as SOS/SOI where back-gate leakage occurs, the simple modeling of leakage current as just one parameter does not yield accurate simulation results.

It was proposed by MRC personnel that at least three different types of leakage currents for each device be accommodated. These currents were defined as channel leakage, edge leakage, and back-gate leakage. The channel leakage is the current passing between the drain and source of a device under the gate oxide. This type of leakage is directly proportional to the width and inversely proportional to the length of the device. The edge leakage is the current passing between the drain and the source around the edges of the channel. This type of current does not depend on the width but is inversely proportional to the length of a given device. The back-gate leakage is the current occurring mostly in SOI/SOS type technologies. The insulator under the channel traps enough charges to create an electric field that support a leakage. This type of leakage is directly proportional to the width and inversely proportional to the length of a device.

For the latest version of PARA, the standby current estimation routine recognizes all these types of leakage currents and calculates the total power requirements for the chip using the relationships mentioned earlier. The main problem encountered for this improvement was the merging of parallel transistors. Sometimes, to increase efficiency, the designer breaks down a big transistor into smaller parallel transistors. The sum of these transistors will have different leakage current than a big transistor due to the dependence of these

leakage currents on total widths and lengths of the devices. The hash table routines were modified to accommodate this problem.

2.2.5 Design of Test ICs

PARA simulation results were compared with results obtained from other accurate, conventional simulators, such as SPICE. The PARA results deviated from the accurate simulation results by <10 percent with three orders of magnitude savings in simulation time for large circuits. However, the simulations results still needed to be verified against experimental data. To accomplish this, a moderately complex IC was designed. The design chosen was a 4-bit parallel multiplier. This design contains around 600 transistors and has total number of possible signal paths in the millions. This layout was first generated using simple rectangular devices with single metal technology. This IC was fabricated by MOS Implementation System (MOSIS) using their n-well, CMOS, nonhardened, 2.0 μ process. The first set of ICs was functionaly operative in pre-irradiation testing and were sent to MRC for radiation exposure. However, it was discovered that the nonhardened technology of MOSIS was capable of only withstanding <100 Rads(Si). With <100 Rads(Si) exposure, the IC showed an extremely high standby current requirement. There were four parts from MOSIS and all showed similar characteristics. Such high leakage per transistor was attributed to edge leakage currents. This lot was fabricated using money provided by MRC.

The next multiplier design was accomplished through the use of donut devices and guard rings. The donut devices are round devices and have no edges between the drain and the source. It was hoped that this would prevent the increase in leakage currents which caused the first set of ICs to fail. However, this set of ICs had a design error in them. The first bit and fourth bit of Y inputs were shorted together through a design error. This error was discovered only after the ICs were fabricated. The verification of the logic system was not carried out due to the inability of the software to recognize donut devices. Berkeley tools were used to extract a device-level description of the circuit but MAGIC is not capable of extracting donut devices and so no logic checks were performed. The cost of the fabrication was not borne by MRC.

The error in the previous version of the design was corrected and the new design sent to MOSIS in late May of 1991. This last version also had an error in it which was not discovered until later. This error seems to have come from transferring the design from one data base to another. At one point, a via contact was transformed into a poly contact. This error was next to the P2 (third least significant bit) output. During simulation, it was found out that P2 is not included in the worst-case path list from PARA, so this error was considered as a noncritical one. The experiment will be continued without any

14

trouble. Also, MOSIS indicated that they can sell 10 additional parts at $15 each. These parts will be ordered and used in the final experiment.

2.2.6   User Interface

Current work is developing a user interface based on the X-Windows system. The interface will be mouse driven and will also incorporate other programs from Vanderbilt. The designers will have pull down menus for ease of use. The first pull down menu will determine the program to be run (i.e., PARA, TODO, PARASTAT, others). For PARA, the second pull down menu will perform the file functions (for example editing, viewing, specifying data files, etc). The third pull down menu will have the simulation results viewing choices (for example, choosing the third most critical dynamic failure characteristics will show the sub-circuit responsible for failure along with bias conditions and delays). The fourth menu will have the test vector generation functions. These menus will be changed as time progresses but their main features will probably remain the same.

2.2.7   Path Properties

It was proposed that path properties for all the worst-case paths be reported to the user. The path properties means the statistics of the path under consideration, such as number of NAND, NOR and inverter gates, a path with maximum number of NAND gates, etc. However, it was later decided that all the information requested above may not be suitable for PARA to calculate. Because PARA operates at the individual device level and not at the logic-level, it was decided to output the characteristic of each gate instead of outputting characteristics of each signal path. These output characteristics of each gate are used for test vector generation also. The output data are stored in a file. The format used is:

```
gate gate_number
output_name
rising_delay falling_delay
minterm input_1 input_2
minterm input_3 input_4
gate gate_number
```

where, gate_number is a number assigned by the designer (or software) and is only for accounting purposes, output_name is the name of the output node for the gate, rising_delay and falling_delay are the delays associated with the gate only. These delays can be viewed as the RC-delay of the gate. The minterms are represented in SOP format

with negation. For the above example, the function of the gate is represented by

$$\overline{\text{input\_1 input\_2}} + \text{input\_3 input\_4}$$

A blank line is the delimiter for the gates (a blank line indicates the end of the definition for a gate). An * at the start of a line indicates a comment line and is to be ignored. This same format is used for the test vector generation routines.

2.2.8 Data Base Accessibility

As the data base with the information that was required by MRC personnel had been removed, this task was not finished.

3.0 CONCLUSION

All the proposed tasks for the PARA project are finished. The data base was modified to accommodate very large circuits for fast simulation. The simulation algorithms were also modified around this data base. A hash table was developed to avoid computationally expensive searches through the data base. Resistive elements were added and leakage current algorithms modified. Overall, the new PARA is ready for distribution after the user interface is completed. However, this will not be a requirement for distribution. PARA can be distributed as it stands now. The main challenge now seems to be the verification of PARA through experimentation. The software will be delivered to MRC in its final form for testing in August 1991. Also PARA will be used for a VLSI Design class at Vanderbilt University during the Fall 1991 semester. This class requires a design and simulation of VLSI-level class project. This will be a good test for PARA to see if it can correctly identify the worst-case signal paths in the circuit. Also, it will be useful to learn how this information is used by the students to improve their designs.

To obtain a copy of PARA, the user simply needs to contact either Dr. Bharat Bhuva or Dr. Shera Kerns at Vanderbilt University. The user will be sent a Vanderbilt University copyright form, for signature, which states that the code will not be distributed outside the requesting organization. The code will then be provided at a nominal handling charge.

APPENDIX A
THE USER MANUAL FOR PARA

This section describes all the necessary commands and the files that the user needs to know about before using PARA. Various input and output options and the file formats are presented. Several example problems are included on the PARA release tape. The files on the tape, too lengthy to reproduce here, illustrate the formats presented.

A.1   GETTING STARTED

PARA can be executed by using one of the following command forms.

        para intInterval circuitFile
        para circuitFile

In the above two cases, the circuitFile refers to the circuit description file. The description is essentially the same as the SPICE description with some modifications and will be discussed in the following section. The intInterval provides the interpolation interval to be chosen for the doses that are present in a file containing transistor parameters at various dose levels. The use of this interval is described in the following sections.

PARA reports all output information on its standard output. In case the output needs to be filed, it can be redirected into the desired output file. The above commands in such cases can be modified to the following:

        para intInterval circuitFile > outputFile
        para circuitFile > outputFile

A.2   INPUT FILES

PARA needs several kinds of input files. These files contain information about the circuit and the parameter shifts of its devices caused by radiation. The various files required by PARA are the following:

        Circuit description file
        .model file
        .res file
        -int.res file
        inverter file

18

The various files are described below.

A.2.1 Circuit Description File

This file contains details of the circuit which PARA is supposed to test. The input file format is the same as the SPICE format with a few additions.

A.2.1.1 Transistor Declaration. As in SPICE, transistors are represented by a string starting with the letter 'm' or 'M'. The rest of the word is immaterial and is ignored by PARA. The string can be used by the user to distinguish between the various transistors. A general form of the transistor declaration is shown below.

MpullUp drain gate source substrate modelName [w = 10u] [l = 2u]
[as = 10e-15] [ad = 10e-15]

where "drain" is the node name of the drain of the transistor, MpullUp; "source" is the node name of the source of the transistor, MpullUp; "gate" is the node name of the gate of the transistor, MpullUp; and "substrate" is the node name of the substrate of the transistor, MpullUp.

All of the above PARA node names are strings; SPICE nodes are designated numerically. The names "VDD" and "vdd" are reserved for the supply voltage. Similarly, "GND" and "gnd" are reserved for the ground. All reserved words can be used either in uppercase or lowercase, but not in a combination of upper and lower cases, i.e., vDD, Vdd, GnD etc., are not allowed.

"modelName" refers to the name of the model associated with the transistor, MpullUp. The model itself can be defined anywhere in the circuit file. All the other variables existing in the declaration are optional. Default values are used if any variables are left unspecified.

An example of the transistor declaration is shown below.

MpassTran output input vdd vdd cmosp w=10u l=2u

A maximum of two lines for the transistor declaration is allowed. The second line should begin with a '+' in the first column in order to represent continuation from the first line. The above example if extended to two lines would look like the following.

MpassTran output input vdd vdd cmosp w=10u + l=2u

A.2.1.2 <u>Resistor and Capacitor Declarations.</u>   A typical resistor declaration starts with an 'r' or 'R' and has the following form:

   RpullUp vdd output 10K

This represents a resistor between "vdd" and "output" nodes with value 10 kΩ. A capacitor has a similar form except that it starts with a 'c' or 'C'.

A.2.2   <u>.inputs CARD</u>

.inputs CARD specifies all the inputs of the circuit. PARA performs the delay calculations only through the paths containing these inputs. Therefore, the user has the flexibility of specifying a part of the circuit for delay calculations. A typical input declaration looks like the following:

   .inputs inp1 clock control1

Here inp1, clock and control1 are the input nodes to be considered.

A.2.3   <u>.model CARD</u>

.model CARD specifies the model to be used with the transistors declared above. The modelNames associated with transistors are defined in the .model CARD. This is similar to SPICE .model card.

A.2.4   <u>.mod File</u>

.mod file contains the SPICE model parameters at each dose level along with other parameters desired for failure calculation. PARA uses this file, along with SPICE and an inverter chain file (described later) to calculate the equivalent ON and OFF resistance values at each dose level. This file should exist in a subdirectory, PARAfiles, and there has to be one .mod file for all fabrication processes. For example, if the given circuit is to be analyzed using the Sandia fabrication process, a file PARAfiles/Sandia.mod should be present. The format for the file is as follows. The first line contains the keyword ".dose" and the corresponding dose value. The second line contains the keyword ".leakage" and the resistances of the n-channel transistors for edge, channel, and back-gate leakage. These resistance values are actually the inverse values of leakage currents for transistors with a W/L ratio of 1. These resistances effectively represent the leakage through the transistors when the node voltage is going HIGH. An example of a .leakage line is given below.

   .leakage 1.0e11 1.0e10 1.0e12

This means that for a transistor with W/L ratio of 1, the edge leakage is 1.0e-11, channel leakage is 1.0e-10 and back-gate leakage is 1.0e-12. All these leakage values are for a supply voltage of 1.0 V.

The next few lines describe the SPICE model parameters for all the desired conditions, i.e., p-channel transistor parameters if it was ON during radiation and OFF during radiation. The same holds for the n-channel transistors. The last line should necessarily begin with an asterisk to mark the end of the dose parameters. After every dose level, a line with the asterisk is essential.

A.2.5   .res File

This file, present in the subdirectory PARAfiles, is created by PARA only if it is not already present. PARA uses the information in the .mod file and runs SPICE on the inverter chain (inverter file is described later) and extracts the resistance values. The ON resistances of n- and p-channel transistors for the particular fabrication process are generated and placed in this file. The initial two lines are the same as in the .mod file but the next line contains "non" "noff" "pon" "poff". "non" and "noff" mean n-channel ON resistance if it was ON during irradiation and n-channel ON resistance if it was OFF during irradiation, respectively. "pon" and "poff" have similar meanings. The last line, which represents the end of all the parameters for the particular dose level starts with an asterisk.

This file can either be created by PARA or by the user simply by editing the file in the appropriate format. Once the file is present in PARAfiles, PARA does not recreate it for other circuits. The same parameters are valid for all the circuits which are to be fabricate ‍ by the same technology. For Sandia.mod, the .res file should be Sandia.res.

A.2.6   -int.res File

In case the user provides the option of interpolation with a specified interpolation interval, PARA creates -int.res file from .res file using linear interpolation. In case the interpolation option is given by the user, PARA creates this file even if it is present in the subdirectory PARAfiles, since the interpolation interval may be different. If the .res file present is Sandia.res, the corresponding file generated is Sandia-int.res. The file can alternatively be generated by the user simply by editing the file using the same format as the .res file.

A.2.7   inverter File

This file is needed at the initial point when .res file is to be generated from the .mod file. This file is to be present and should always be named as invchain. PARA runs SPICE on

21

the inverter file to generate the various resistances using different models in the .mod file. This essentially means that for the first run on a circuit with a new technology, PARA is going to be slow, since SPICE is to run for all the sets of model parameters in the .mod file. All other runs, on different circuits but using the same technology (fabrication process) will take less time. The inverter file is always the same and is provided with PARA. Also, at the beginning, the user should make sure that SPICE can be run from the directory where PARA resides, i.e., an executable version of SPICE is to be present in the directory or an appropriate path should be set.

A.2.8 Output Files

PARA does not generate an output file directly. All the output information is printed on the standard output. In order to file the output, redirection of the output to a file can be requested. PARA then prints out the node degradations, path degradations, and power-supply degradation for each dose level. The percentage variation from the zero dose level is specified and it is left to the user to determine if the degradations are large enough for the failure to occur.

A.2.9 Input Options

Several options are available based on the kind of data files that are available. During the first run on a circuit using a particular technology, the .res file is normally not present. The user therefore has the flexibility of either providing the .mod file which can be used by PARA to generate the .res file or editing the file manually filling all the data required in the .res file. PARA checks to see if the .res file exists in the PARAfiles directory. If the file is present, then PARA uses it directly, else PARA first tries to generate the file using SPICE, .mod file, and the inverter file, and then uses it on the input circuit. For all the subsequent runs on different circuits, but using the same technology (e.g.Sandia), it uses the .res file generated.

In case the -int.res file is present in the directory PARAfiles, this file is given preference over .res file; i.e., -int.res file is used rather than .res file. The user has the flexibility of providing the interpolation intervals as desired. Whenever the interpolation intervals are provided in the command line, the PARA uses the interpolation on .res file to generate -int.res file regardless of whether -int.res file is already present in PARAfiles. Therefore, if the user wants to preserve a number of -int.res files with different interpolation intervals, he or she needs to rename the files with different file names before providing the new interpolation interval to PARA. In case the -int.res file is absent and no interpolation option is given in the command line, the .res file would be used. As with the other files, the user can edit the -int.res file himself, keeping in mind the appropriate format. By

22

the interpolation options and the combination of various files, a number of input options are possible.

A.2.10   Limitations

There are some restrictions on the kind of circuits that the current version of PARA can handle. Only CMOS Digital circuits can currently be analyzed. These include all the gates, latches and pass transistors (transmission gates). The circuits that are out of the PARA domain are the ones that do not have an equal number of complementary p- and n-channel transistors. Any form of analog circuitry which may occur in digital circuits, including sense amplifiers and dynamic RAMs, are not handled. External resistances are not handled in the current version. In addition, the capacitances at nodes other than signal nodes are ignored.

PARA provides accurate delay values for most cases. For cases when a very slow element is present in series with a sequence of fast elements, PARA does not provide very accurate delay values, but correctly identifies the critical path. As in any other switch-level simulator, PARA sums up the delays at individual gate nodes to obtain the total value. In actuality, two consecutive gates have an overlap of real time during which they approach their final value. Therefore, the addition of delays to get the total delay is not precise. The error introduced by this summation is very small for most cases, except for the circuit paths having extremely slow elements. Such paths are therefore not handled properly by PARA. PARA provides higher delay values in such cases and hence is pessimistic in nature. This situation can be seen more clearly, if it is assumed that the external capacitance at a node in a circuit being simulated has a value 10 uF instead of 10 pF. In this case, that node rises to a HIGH level (5 V, say) very slowly as compared to other nodes. As it rises to 2.5 V, it starts affecting the next gate and therefore the output of that gate starts falling to LOW level simultaneously. The possibility of this output node reaching the LOW level before original reaches the HIGH level cannot be ruled out. This clearly indicates that the total delay in this case cannot be the sum of the delays at these nodes. PARA does add up the delays and provide a higher value of total delay.

PARA carries out a linear interpolation of the parameters for the intermediate dose values. This is not very accurate if the data points available are far apart. Accurate results are expected if the parameters in .res file are available at small dose intervals.

PARA does not simulate failures due to field-oxide leakage.

A.3   SOURCE CODE FILES

Important source code files are briefly described here.

A.3.1 Main File

The main file that controls the sequence of operations and contains the main function is para.c.

A.3.2 Include Files

These files are included in most of the other source code files. They contain all the data declarations and constant definitions.

A.3.2.1 constant.i. This file defines all the constants used in all .c files.

A.3.2.2 struc.i. All the structures for devices, capacitances, signal nodes and signal paths are defined here.

A.3.2.3 def_str1.i & def_str2.i. These define some of the default values for device parameters and are easily changeable.

A.3.3 Input Interface Files

These files control the front end and interact with SPICE for the extraction of relevant parameters.

A.3.3.1 resistors.c. The ON resistance values of the devices are calculated by this file.

A.3.3.2 modifyres.c. This function in this file changes the resistances of the devices after each run for a dose.

A.3.3.3 interpol.c. Interpolation of the resistance values is carried out by the function defined in this file.

A.3.4 Data Base Generator Files

These files parse the input circuit description file and generate the data base for subsequent failure simulations.

A.3.4.1 getdevice.c. All the resistor and capacitor declarations are scanned and placed into appropriate data structure.

A.3.4.2 getr.c. The transistor declarations are read and stored.

A.3.4.3 <u>putmodel.c</u>. The models corresponding to the transistors stored are attached to the transistor descriptions.

A.3.4.4 <u>inputs.c</u>. .inputs card is read and the input nodes are stored for subsequent signal flow generation.

A.3.4.5 <u>getnode.c</u>. All the signal nodes are extracted and the transistors associated with them are stored along with them.

A.3.4.6 <u>sig_cap.c</u>. The node capacitances are calculated and attached to the description of the corresponding node.

A.3.4.7 <u>sigflow.c</u>. The signal paths are generated for dynamic failure calculations.

A.3.5 <u>Failure Detection Files</u>

The failure simulation algorithms are implemented in these files.

A.3.5.1 <u>minpath.c</u>. The n-channel and p-channel resistances for static failure determination are calculated in this file. Power supply related failure is also handled in this file.

A.3.5.2 <u>delay.c</u>. Dynamic failure calculations are done by the functions defined in this file. The individual delays at each nodes are calculated and added to form the total delay.

A.3.6 <u>Other Files</u>

These files do not fall into any special category.

A.3.6.1 <u>loads.c</u>. This file loads a single line from a file into a buffer.

A.3.6.2 <u>getword.c</u>. This file allows a word to be loaded into a buffer from a line stored in another buffer.

A.3.6.3 <u>makefile</u>. This file generates PARA for execution.

APPENDIX B
TEST VECTOR GENERATION FOR WORST-CASE TESTING
OF CMOS ICs IN TOTAL DOSE ENVIRONMENT

B.1   INTRODUCTION

This appendix presents algorithms along with experimental verification to automatically generate input vectors to verify the radiation tolerance for a given CMOS IC. The tolerance is estimated through worst-case bias conditions to induce maximum shift in circuit parameters causing earliest operational failure. The software generates the global input test vectors to induce these worst-case bias conditions at individual signal nodes for testing the ICs.

It has been shown that the operational failure of ICs in total dose environments is bias-dependent. The circuit-level parameters causing failure (such as delay, power, etc) are dependent on individual transistor characteristics which, in turn, are bias- and radiation-dependent. With bias-dependent transistor parameter shifts, it is possible that for some bias conditions the circuit parameters will exhibit greater degradation. For example, for an inverter, if the input is held HIGH (or LOW) during irradiation, p-channel devices will experience shifts corresponding to that for an OFF (or ON) device and n-channel devices will experience shifts corresponding to that for ON (or OFF) device. This inverter will exhibit higher gate delay and power for the case where input was held HIGH than for the case where input was held LOW due to higher p-channel threshold voltage and n-channel leakage. Thus, bias-dependency of the device parameters shifts indicates the existence of worst-case bias conditions for which the degradation in circuit performance is greater than other bias conditions. For accurately estimating the operational life of an IC, these worst-case conditions must be used to test for the earliest operational failure.

However, usually the test engineers use arbitrary input vectors to test a given design. This will not yield an accurate estimation of the radiation tolerance of the given ICs. Also, in comparing two different designs for the same function (internal designs of the ICs are different but they perform the same function), the use of identical test vectors may not yield an accurate estimation of the operational lifetime. To compare two different parts, one must use individual worst-case test vectors. If the test vector chosen happens to be worst-case for design A and not for design B, then design B may exhibit better radiation

tolerance. However, design A may prove to be better if individual worst-cases were used for testing.

The task of identifying the worst-case test vectors can be divided into two subtasks; (1) identification of worst-case bias conditions for individual transistors in the circuit, and (2) identification of worst-case test vectors based on these bias conditions. For the first task, software which has the capability of identifying the worst-case bias conditions for individual transistors using switch-level simulation along with RC-delay estimation techniques has been developed. For the second task, software capable of using back-tracking and back-tracing techniques to identify the worst-case test vectors for a given design has been developed. This report deals mainly with the software algorithms for the second task and the results of the experiments carried out to verify these algorithms.

B.2 ALGORITHMS

The software developed for the first task identifies the worst-case bias conditions for all devices responsible for causing the failure. For a given circuit, the number of critical devices or the size of the critical sub-circuit is usually much smaller than the whole circuit. As only the worst-case for the critical sub-circuit is known, the bias conditions still have to be identified for the rest of the devices and these conditions extended to obtain a valid test vector for testing. For example, for an arbitrary circuit with worst-case bias conditions for delay testing as shown in Figure B-1, the node voltages at nodes A, B, C, D, and E must be identified to induce the worst-case conditions for testing. Looking at the design, it is evident that the node F must be LOW to cause output node to remain HIGH. To obtain a LOW at node F, node E or node D or both must be at HIGH. As only the critical sub-circuit (consisting of logic gates 1, 2, 3, and 4) is of interest, any one of the above three choices for node E and D will induce the required worst-case conditions for critical sub-circuit. This process of obtaining unknown node voltages can be repeated until all node voltages are known to identify a test vector. Such a process can be performed manually for small circuits, but for large circuits, computer-aided test vector generation techniques must be used.

The software developed to generate such test vectors for large CMOS circuits use back-tracing and back-tracking techniques. During back-tracing techniques, the software propagates the worst-case conditions from output towards inputs just as in the above example from output to F to E/D nodes. This back-tracing technique alone is not enough to obtain the test vectors as conflicts will arise. A conflict arises when a single node is forced to have a HIGH and a LOW value at the same time. For example, for an arbitrary circuit in Figure B-2, to induce the worst-case, the output HIGH logic level was traced to a LOW for node G. And subsequently, a HIGH for node B. However, a HIGH at node F requires that node B be assigned a LOW. This is a conflict. For such a case, the
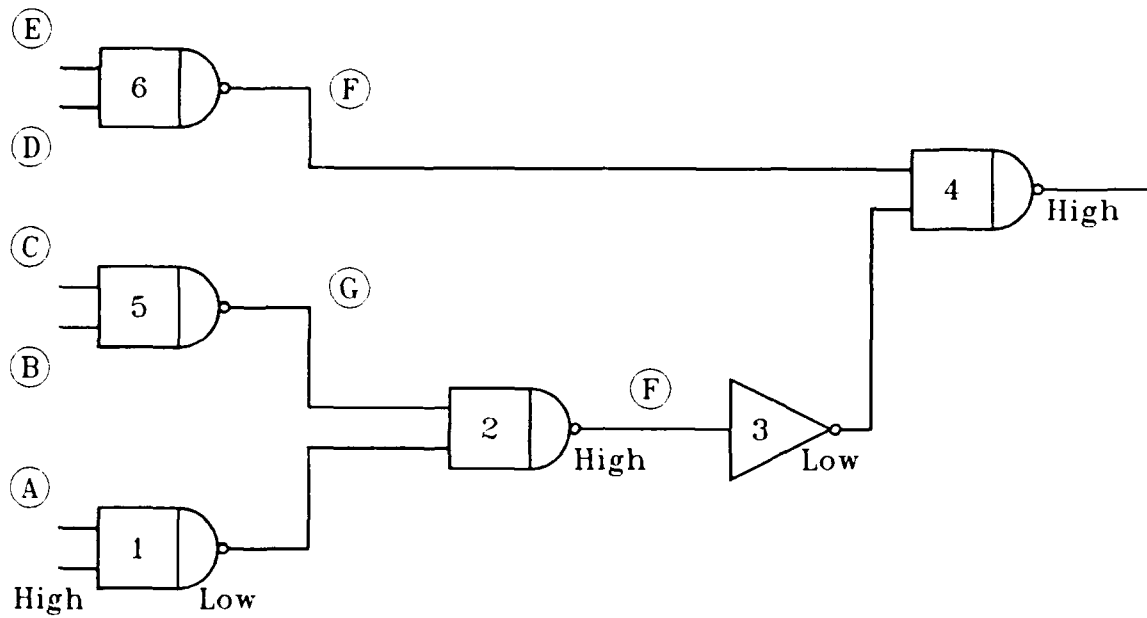
27

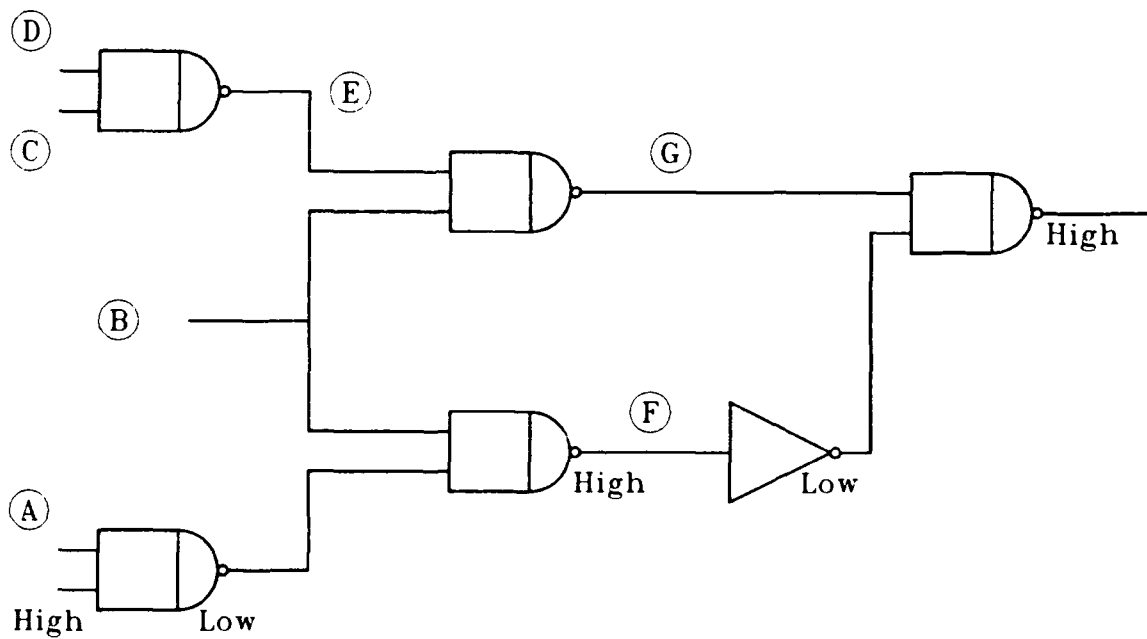Figure B-1. A simple example for test vector generation.

Figure B-2. A simple example to show conflict in test vector generation.

28

assignment would be a LOW for B, a HIGH for E, and LOWs for C and D. For this simple example, the conflict resolution was easy. However, for complex circuits, such manual resolution may not be possible. For such cases, use back-tracking techniques. During back-tracking, go back toward an output node until another assignment for the conflicting node can be found. For example, propagate forward in the above example after conflict at node B is found to reach node G. To obtain a LOW at this node, there can be an alternative assignment for node B, namely a HIGH for node E and a LOW for B. Then, this assignment is propagated using back-tracing as discussed above until another conflict is found. At that time the whole process of back-tracking and back-tracing is repeated.

B.3 RESULTS

The test vector generator software has been tested on many designs including 4-bit carry look-ahead, 4-bit multiplier, Binary-Coded-Decimal and many other designs. In all cases, the software found the worst-case paths and the test vectors to induce the worst-case degradations. The simulation results were verified through circuit simulation from conventional simulators, such as SPICE. The results obtained from SPICE exactly matched those obtained from these simulators. However, the simulation time for each circuit was reduced by two to three orders of magnitude. For example, carry-look-ahead example took less than 2 minutes of CPU time on a SUN 3/50 while SPICE took hours along with user interventions to obtain the same results. The simulation time for multipliers was measured in days for SPICE while this software takes <1 minute for each test vector.

B.4 CONCLUSIONS

An Automatic Test vector Generation procedure has been developed to compute the vectors necessary for generating the worst-case conditions for an IC for testing it against earliest failure in a total dose environment. Such a software is necessary for accurately estimating the radiation tolerance of a given design and for comparing two different designs representing the same logic function.

APPENDIX C
EVALUATION OF PARA PREDICTIONS

C.1   INTRODUCTION

An independent, experimental verification of PARA predictions was planned as part of the development effort documented in this report. A 4-bit multiplier designed by the staff at Vanderbilt University was the vehicle selected for the verification demonstration. The plan was to fabricate the design through MOSIS (MOS Implementation System), and then perform pre-irradiation and post-irradiation characterization of test transistors and multiplier circuits contained on the same die. The transistor characteristics were to be used to parameterize the PARA model to predict worst case propagation paths. The pre- and post-irradiation propagation delays for the multiplier were measured as a function of input vector to determine empirical values. The intent was to compare the PARA predictions with the empirical results in terms of the value of the propagation delay and the test vector ranking for worst case propagation delay.

C.2   SUMMARY OF RESULTS

The empirical verification of PARA was not achieved. Problems were encountered in both the experimental and analytical portions of the effort. They are summarized in the following paragraphs.

C.2.1   Initial Multiplier Lot

The first lot of multipliers was fabricated by MOSIS and resulted in eight packaged parts. The parts were electrically characterized at MRC, and five units were found to be appropriate for testing. Several of the MOS test transistors were found to have ruptured gates, presumably from electrostatic discharge (ESD) damage during packaging. The parts were shipped in conductive foam and ESD handling precautions were taken at Vanderbilt and MRC. In the five good units, there were enough test transistors to provide "on" and "off" bias conditions for both NMOS and PMOS transistors. Typical drain characteristics for NMOS and PMOS devices are shown in Figures C-1 and C-2, respectively. The multipliers were found to be functional, but they were extremely sensitive to latchup. However, with careful handling (i.e., input current limited by 10 kΩ resistors and power supply current limited by 210 kΩ resistor), they could be tested to determine propagation delays as a function of test vector. The results of these measurements for unit 7 are shown in Figure C-3.

30
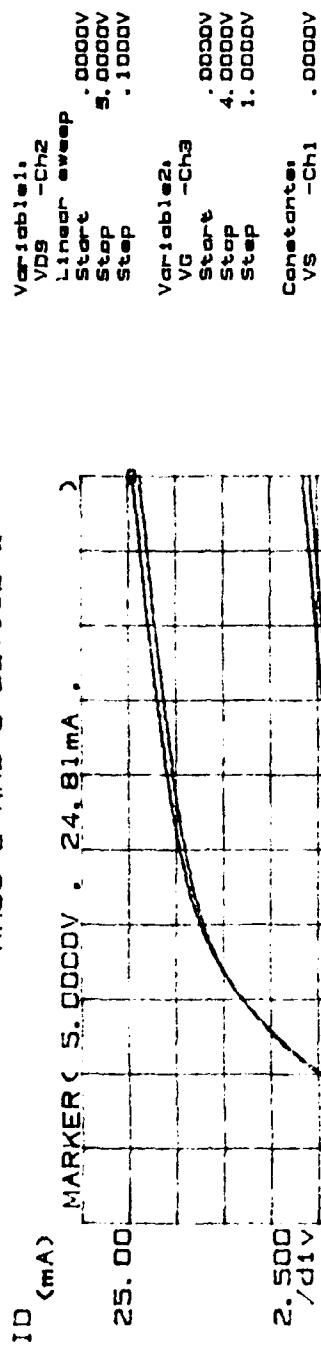
Figure C-1. Typical NMOS drain characteristic.

****** GRAPHICS PLOT ******
PMOS 2 AND 3 DEVICE 2

ID (mA)

MARKER (-5.0000V , -10.68mA ,)

-15.00

1.500
/div

.0000
.0000

VDS   .5000/div   (V)   -5.000

PMOS2
PMOS3

Variable1:
VDS   -Ch2
Linear sweep
Start   .0000V
Stop   -5.0000V
Step   -.1000V

Variable2:
VG   -Ch3
Start   .0000V
Stop   -4.0000V
Step   -1.0000V

Constant:
VS   -Ch1   .0000V
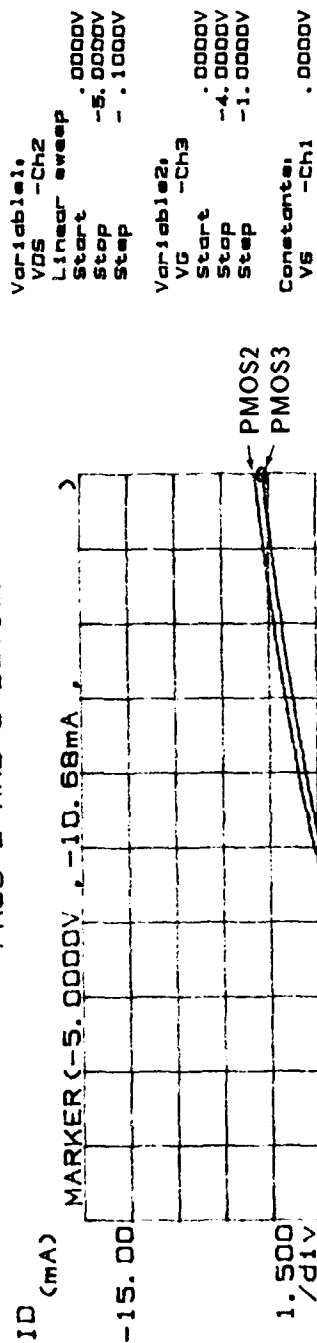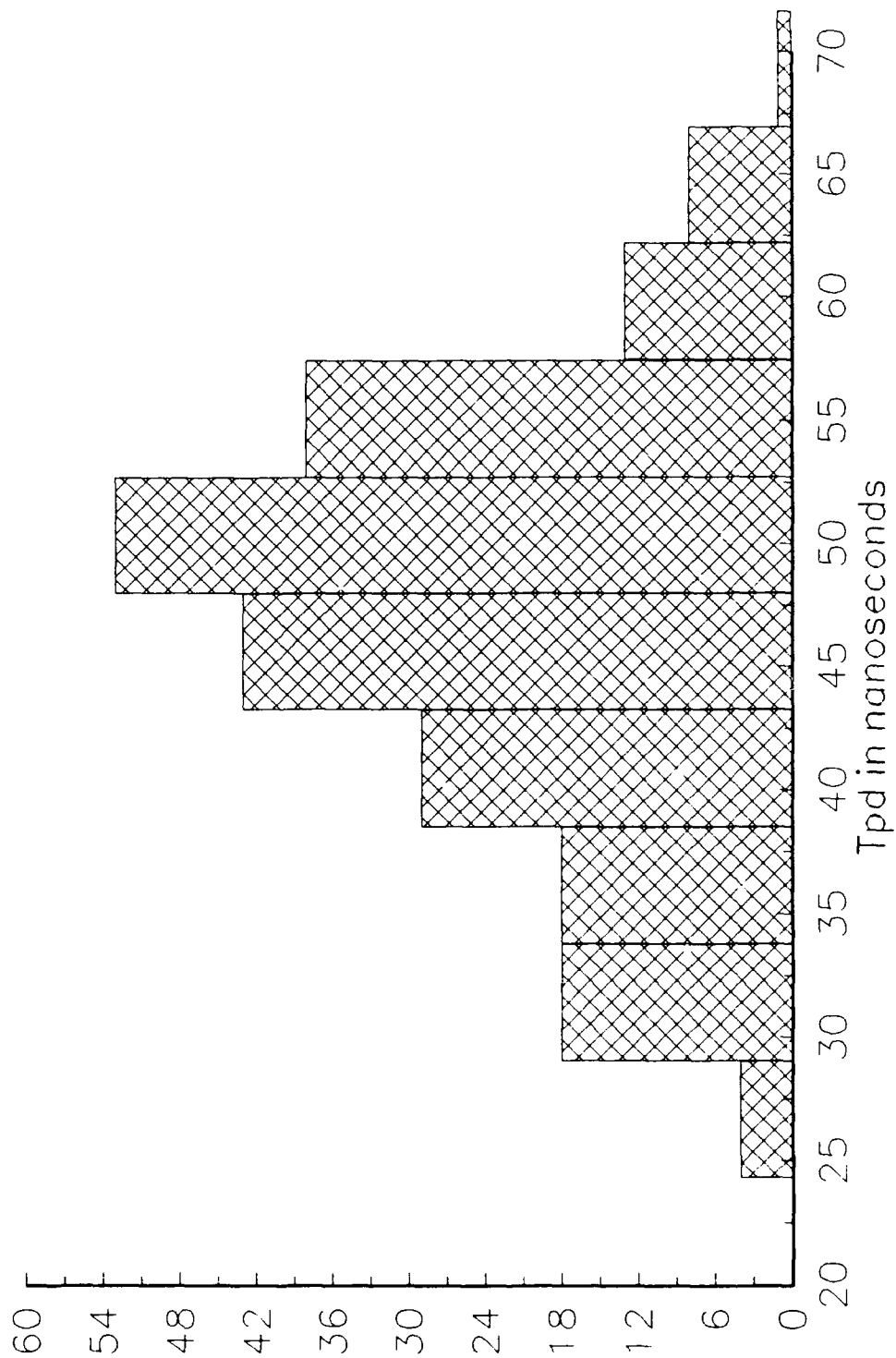
Figure C-2. Typical PMOS drain characteristic.

Figure C-3. Distribution of pre-irradiation propagation delay.

Irradiations were conducted at the Phillips Laboratory industrial X-ray source with an energy of 20 keV and a dose rate of 600 Rad(Si)/min. The first device (unit 7) was found to have failed when interrogated after 1 minute of irradiation. Failure was due to power supply current increasing to 12 μA (pretest value was 20 nA) and consequently dropping too much voltage across the current limiting resistor. Unit number 8 was then tested at a reduced energy (15 keV) and dose rate (80 Rad(Si)/min) and monitored continuously during the irradiation. The device was fully functional after 45 seconds of irradiation. At that time, the supply current began a monotonic increase, and the device failed after 1.3 minutes due to excessive supply current. The mechanism for the failure was determined to be source-to-drain edge leakage as shown by the subthreshold characteristic of the irradiated NMOS transistor in Figure C-4. With the body and source of the transistor at the same potential, the subthreshold current has increased over five orders of magnitude from its pre-irradiation value. If the body voltage is increased to 2.5 V, the pre-irradiation value of the subthreshold current can be maintained. This behavior is symptomatic of radiation induced edge leakage.

The failure from edge leakage at such low doses precluded any evaluation of total dose induced propagation delay variations. Although edge leakage had been anticipated as a problem, it was not expected to cause failure until 10 krad to 100 krad. Unfortunately, the commercial MOSIS process was softer than anticipated, and the experimental effort was terminated.

An attempt was made to test the remaining devices at elevated temperature to produce a change in transistor characteristics and related change in propagation delay. The high temperature transistor characteristics could have been used as inputs to the PARA model to predict high temperature path delays. Unfortunately, the devices were so susceptible to latchup at elevated temperatures that power could not be applied without latchup occurring. No further effort was expended on this lot to conserve program resources.

C.2.2   Hughes 8-Bit Multiplier

An attempt was made to locate a radiation hardened device which could be simulated with PARA and radiation tested. A Hughes 8-bit multiplier was identified, and eight sample devices were acquired from U.S. Army Harry Diamond Laboratory. A schematic was acquired from Hughes and sent to Vanderbilt. Unfortunately, PARA could not simulate the Hughes multiplier, because it used a noncomplementary, 2-phase clocked gate design. Modifying PARA to handle such designs would have been a major change. Therefore, no additional effort was expended on the Hughes multiplier. All other radiation hardened microcircuits considered were too large for simulation.
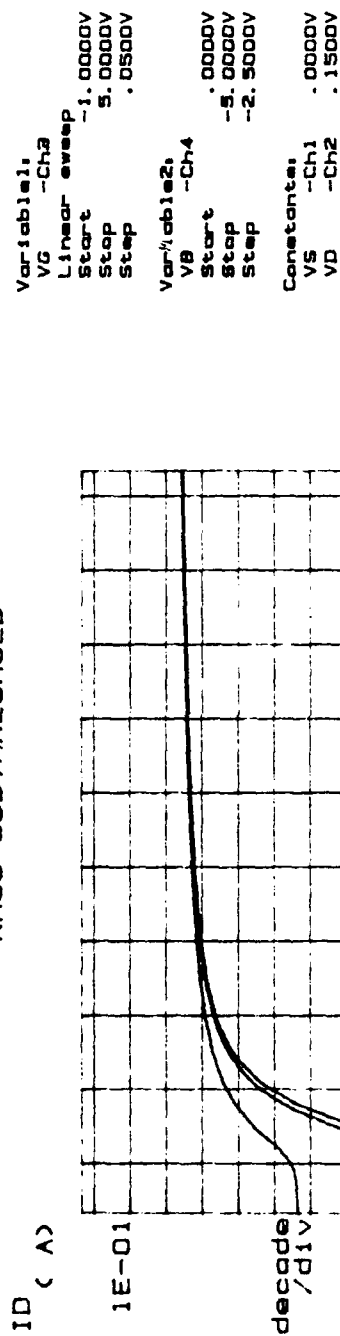
Figure C-4. Radiation-induced edge leakage.

C.2.3 Second Multiplier Lot

Since radiation-induced edge leakage was the failure mechanism for the first lot, Vanderbilt personnel redesigned the multiplier to use closed geometry transistors (i.e., drains surrounded by annular gate and source) to eliminate the source-to-drain edge. Channel stop implants were also used in the P-wells to reduce leakage under the field oxide. The input protection structures were redesigned to reduce latchup susceptibility. Unfortunately, there was a design error which caused the lot to yield zero.

C.2.4 Third Multiplier Lot

Vanderbilt personnel corrected the design error from the second lot and submitted a new low to MOSIS. A new design error caused the P2 output to be disconnected. No propagation path involving P2 could be evaluated. However, 19 devices were acquired to support testing in case the PARA predictions indicated worst case paths that did not involve P2. The verification would be partially compromised since the effect of P2 on propagation delay could not be observed. However, the measured and predicted values of delay for other test vector/output combinations could be compared. These measurements were not attempted for the reasons noted in the following paragraph.

C.2.5 PARA Predictions for 4-Bit Multiplier

PARA's predictions are made for connectivity paths as defined in the netlist. It identifies bias conditions at nodes which would result in worst case path delay. It does not consider whether or not the bias conditions can be physically achieved. Neither does it consider whether or not the path can physically propagate a signal. A test vector generation program is required to identify false paths and indicate the input vectors corresponding to the worst case physical path. The PARA path predictions were used as inputs to a test vector generation program developed at Vanderbilt. The initial attempt considered the 8000 worst paths, and a subsequent attempt considered 80,000 paths. No physically realizable propagation path could be found by the test vector program. It is unclear whether PARA or the test vector program is at fault. There were insufficient resources left to identify and correct the problem. Since the object of the experimental effort was to verify PARA predictions, no additional effort was expended.

C.3 CONCLUSIONS

The demonstration of the accuracy of PARA in predicting worst case test vectors and quantitative post-irradiation propagation delay remains the key issue in its application to radiation hardened design and testing problems. The pacing item is the development of a method to yield valid test vectors from the PARA path information. If this can

36

be done, the design error in the 4-bit multiplier could be corrected and a new MOSIS lot fabricated for less than $1000. The test instrumentation was fully developed on this program. A complete test could be fielded and the data reduced within two weeks. Thus, the verification effort is minimal, but the effort required for the PARA/test vector solution is unknown at this time.